Rapport partie qui ont sauté :

Les diagrammes d'activité qui vont suivre vont permettre de détailler les différentes étapes possibles du programme. La partie en vert représente les actions de l'administrateur tandis que la partie en bleu représente les actions de l'utilisateur.

Les 3 diagrammes qui suivent ont la première partie identique. Cette partie correspond à la mise à jour des données.

Le diagramme d'activité qui suit illustre le déroulement du programme lorsque l'utilisateur choisit la requête 1. Dans ce cas-là, il veut trouver les communes ayant des parcelles limitrophes à une commune qu'il aura choisi.

\\

Dans le cas de la requête 1, l'utilisateur veut trouver les communes ayant des parcelles limitrophes à une commun

Le programme va demander à l'utilisateur la commune qu'il veut étudier, et celle-ci représentera l'entrée. Le programme s'assuera au préalable que la commune existe, ce qui permet de renvoyer directement une erreur à l'utilisateur si cette condition n'est pas vérifiée.

\\

Ensuite, le programme doit seulement lire la base de données qui est déjà construite afin de trouver les communes adjacentes à celle qui a été sélectionnée. Le résultat est retourné à l'utilisateur sous forme de sortie Json.

\\

\begin{figure}[H]

\centering

\includegraphics[scale=0.5]{Requête 1 (4).pdf}\\

\captioCastextbf{Diagramme d'activité requête 1.}

\end{figure}

\newpage

Le diagramme d'activité qui suit illustre le déroulement du programme lorsque l'utilisateur choisit la requête 2. Dans ce cas-là, il veut trouver les parcelles en limite d'une commune qu'il aura choisi.

\\

L'utilisateur fournira en entrée l'identifiant de la commune pour laquelle il veut connaitre les parcelles en limites. Le programme s'assuera également au préalable que la commune existe, ce qui permet de renvoyer directement une erreur à l'utilisateur en cas de problème.

\\

Pour répondre à cette question, le programme charge les parcelles de la commune sélectionnée, et va comparer les points de coordonnées de la communes et de toutes les parcelles contenues dans celle-ci. Si une parcelle et la commune partagent un segment en commun, alors la parcelle est en limite de la commune et est donc stockée. Le programme s'arrête quand tous les points ont été comparé et retourne donc le résultat dans un fichier Json.

\begin{figure}[H]

\centering

\includegraphics[scale=0.5]{Requête 2 (1).pdf}\\

\captioCastextbf{Diagramme d'activité requête 2.}

\end{figure}

\newpage

Le diagramme d'activité qui suit va illustrer le déroulement du programme pour la requête 3. Dans ce cas-là, l'utilisateur veut trouver les parcelles contigües à une parcelle qu'il aura choisi.

\\

Le programme va demander à l'utilisateur la parcelle dont il cherche les parcelles contigües, et celle-ci représentera l'entrée. De la même manière, le programme s'assure qu'il n'y a pas d'erreur.

\\

Avant de charger les données, le programme va vérifier si la parcelle sélectionnée est en limite d'une commune. Autrement dit, l'exécution de cette requête va aussi passer par l'exécution de la requête 2 qui va chercher si la parcelle sélectionnée est en limite d'une commune. Grâce à l'identifiant de la parcelle on peut récupérer la commune à laquelle elle appartient, et donc le programme pourra exécuter la requête 2. Cette stratégie permet de ne pas parcourir trop de points inutilement. De plus, si l'utilisateur a déjà exécuté la requête 2 au préalable, le résultat 2 est déjà connu.

Si la parcelle est en limite de la commune, le programme chargera les parcelles de sa commune et des communes adjacentes, sinon la parcelle n'est pas en limite et donc le programme chargera uniquement les parcelles de la commune.

Ensuite, de la même manière que dans les requêtes précédentes, le programme regardera s'il y a des segments en commun et les stockera. Une fois que tous les segments sont parcourus, le programme retournera le résultat dans un Json.